PUC Minas

Algoritmos e Estruturas de Dados

Edwaldo Soares Rodrigues
Departamento de Ciência da Computação
PUC Minas São Gabriel
2023/2

Recursão

- Na linguagem C#, uma função pode chamar uma outra função;
  - A função Main() pode chamar qualquer função, seja ela da biblioteca da linguagem (como a função Console.WriteLine()) ou definida pelo programador (função imprime());
- Uma função também pode chamar a si própria;
  - A qual chamamos de ***função recursiva***;

Recursão

- A recursão também é chamada de definição circular. Ela ocorre quando algo é definido em termos de si mesmo;

- Um exemplo clássico de função que usa recursão é o cálculo do fatorial de um número:

  - 3! = 3 * 2!
  - 4! = 4 * 3!
  - n! = n * (n - 1)!

Recursão

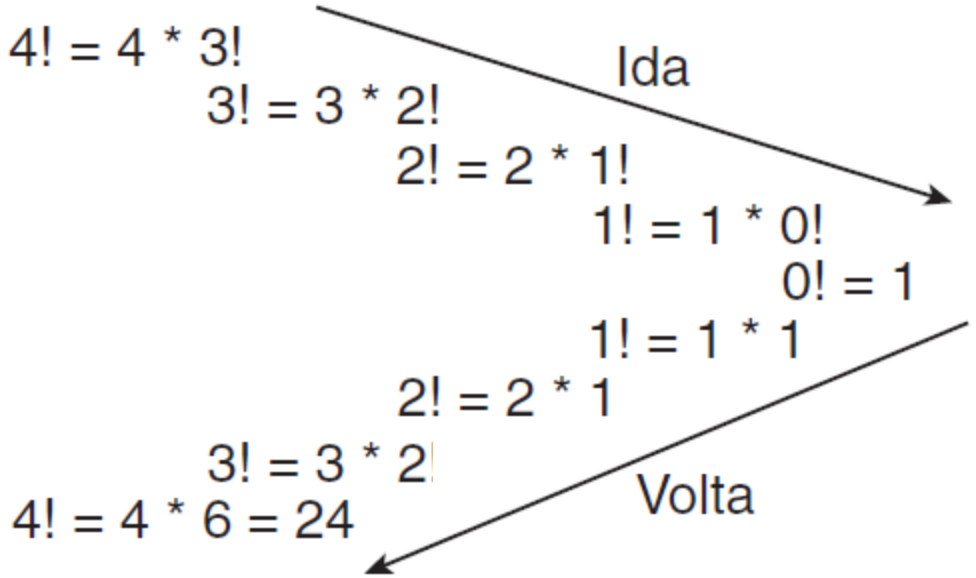
0! = 1

1! = 1 * 0!

2! = 2 * 1!

3! = 3 * 2!

4! = 4 * 3!



n! = n * (n - 1)! : fórmula geral

0! = 1 : caso-base

PUC Minas

Recursão

- Uma função recursiva apresenta dois componentes básicos:
  - Caso base (critério de parada);
  - Fórmula geral (vai permitir que uma função chame a si mesma);
- Identificar estes dois componentes é fundamental, quando se deseja criar um algoritmo recursivo;

PUC Minas

Recursão

Com Recursão

```java
public static int fatorialR(int n){
  if(n == 0){
    return 1;
  }
  else{
    return n * fatorialR(n-1);
  }
}
```

Sem Recursão

```java
public static int fatorial(int n){
  if(n == 0){
    return 1;
  }
  else{
    int i, f = 1;
    for(i = 1; i <= n; i++){
      f *= i;
    }
    return f;
  }
}
```

Recursão

```
using System;

class Program{

  public static int fatorialR(int n){
    if(n == 0){
      return 1;
    }
    else{
      return n * fatorialR(n-1);
    }
  }

  public static int fatorial(int n){
    if(n == 0){
      return 1;
    }
    else{
      int i, f = 1;
      for(i = 1; i <= n; i++){
        f *= i;
      }
      return f;
    }
  }

  static void Main(string[] args){
    int r = fatorial(5);
    int r2 = fatorialR(5);
    Console.WriteLine("Fatorial = " + r);
    Console.WriteLine("Fatorial = " + r2);
  }
}
```

Recursão

- Em geral, formulações recursivas de algoritmos são frequentemente consideradas "mais enxutas" ou "mais elegantes" do que formulações iterativas;

- Porém, algoritmos recursivos tendem a necessitar de mais espaço do que algoritmos iterativos;

PUC Minas

Recursão

- Todo cuidado é pouco ao se fazer funções recursivas;
  - Critério de parada (caso base): determina quando a função deverá parar de chamar a si mesma;
  - O parâmetro da chamada recursiva deve ser sempre modificado, de forma que a recursão chegue a um término;

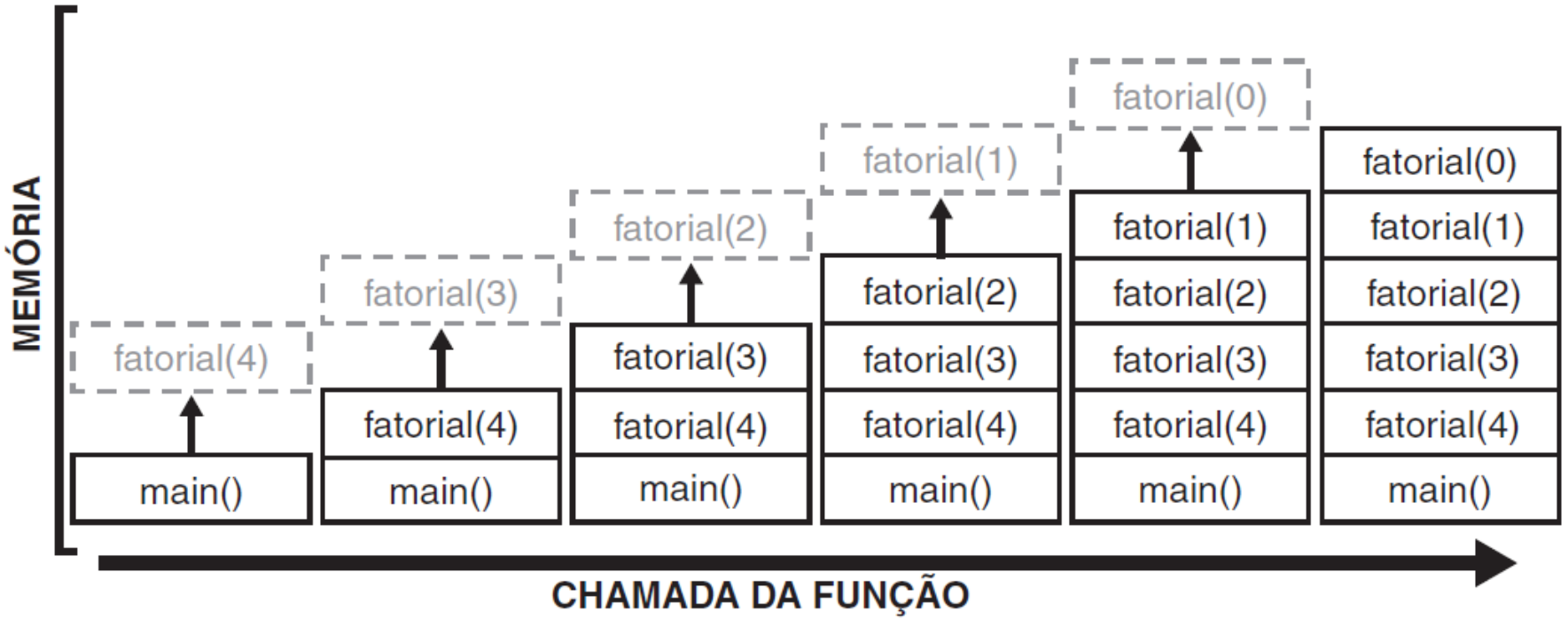
```
public static int fatorialR(int n){
  if(n == 0){ //critério de parada
    return 1;
  }
  else{ //parâmetro passado para a função sempre muda
    return n * fatorialR(n-1);
  }
}
```

PUC Minas

Recursão

- O que acontece na chamada da função fatorial com um valor como n = 4?

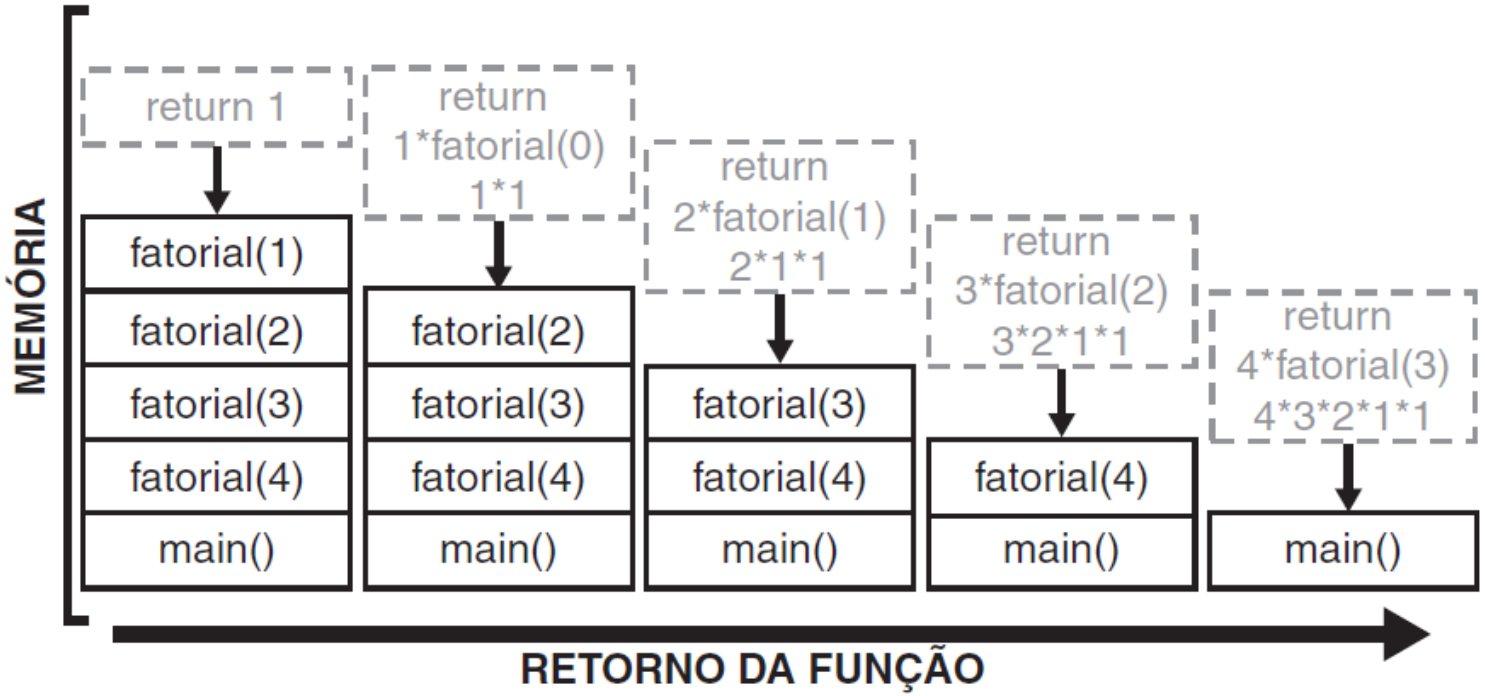
```
int x = fatorial(4);
```



PUC Minas

Recursão

- Uma vez que chegamos ao caso-base, é hora de fazer o caminho de volta da recursão;



PUC Minas

Fibonacci

- Essa sequência é um clássico da recursão
  - 0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, ...

- A sequência de Fibonacci é definida como uma função recursiva utilizando a fórmula a seguir;

$$F(n) = \begin{cases} 0, & \text{se n = 0} \\ 1, & \text{se n = 1} \\ F(n-1) + F(n-2), & \text{outros casos} \end{cases}$$

- Sua solução recursiva é muito elegante ...

PUC Minas

Recursão

Com Recursão

```java
public static int fiboR(int n){
  if(n == 0 || n == 1){
    return n;
  }
  else{
    return fiboR(n-1) + fiboR(n-2);
  }
}
```

Sem Recursão

```java
public static int fibo(int n){
  int i, c, a = 0, b = 1;
  for(i = 0; i < n; i++){
    c = a + b;
    a = b;
    b = c;
  }
  return a;
}
```

PUC Minas

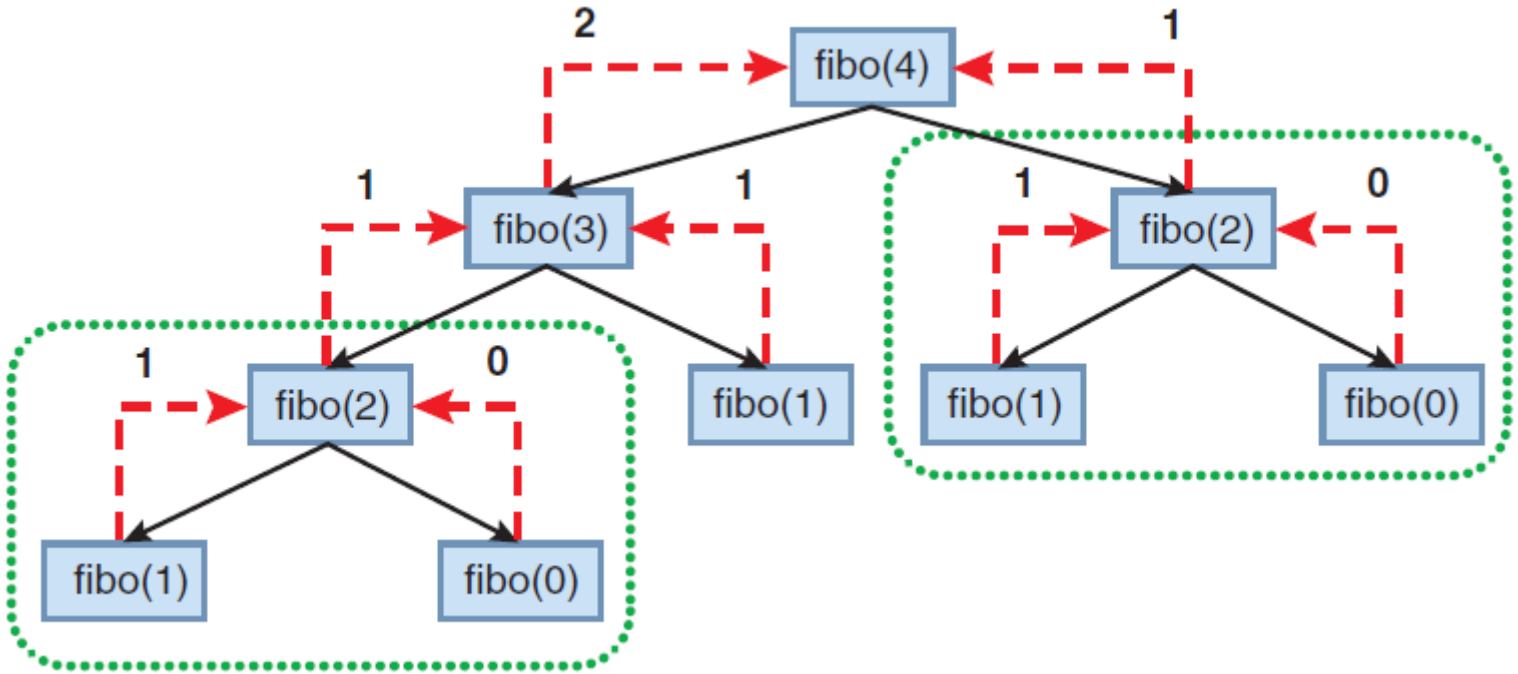
Recursão

```
using System;

class Program{
  public static int fiboR(int n){
    if(n == 0 || n == 1){
      return n;
    }
    else{
      return fiboR(n-1) + fiboR(n-2);
    }
  }

  public static int fibo(int n){
    int i, c, a = 0, b = 1;
    for(i = 0; i < n; i++){
      c = a + b;
      a = b;
      b = c;
    }
    return a;
  }

  static void Main(string[] args){
    int f = fibo(7);
    int fr = fiboR(7);
    Console.WriteLine("Fibonacci = " + f);
    Console.WriteLine("Fibonacci = " + fr);
  }
}
```

PUC Minas

Fibonacci

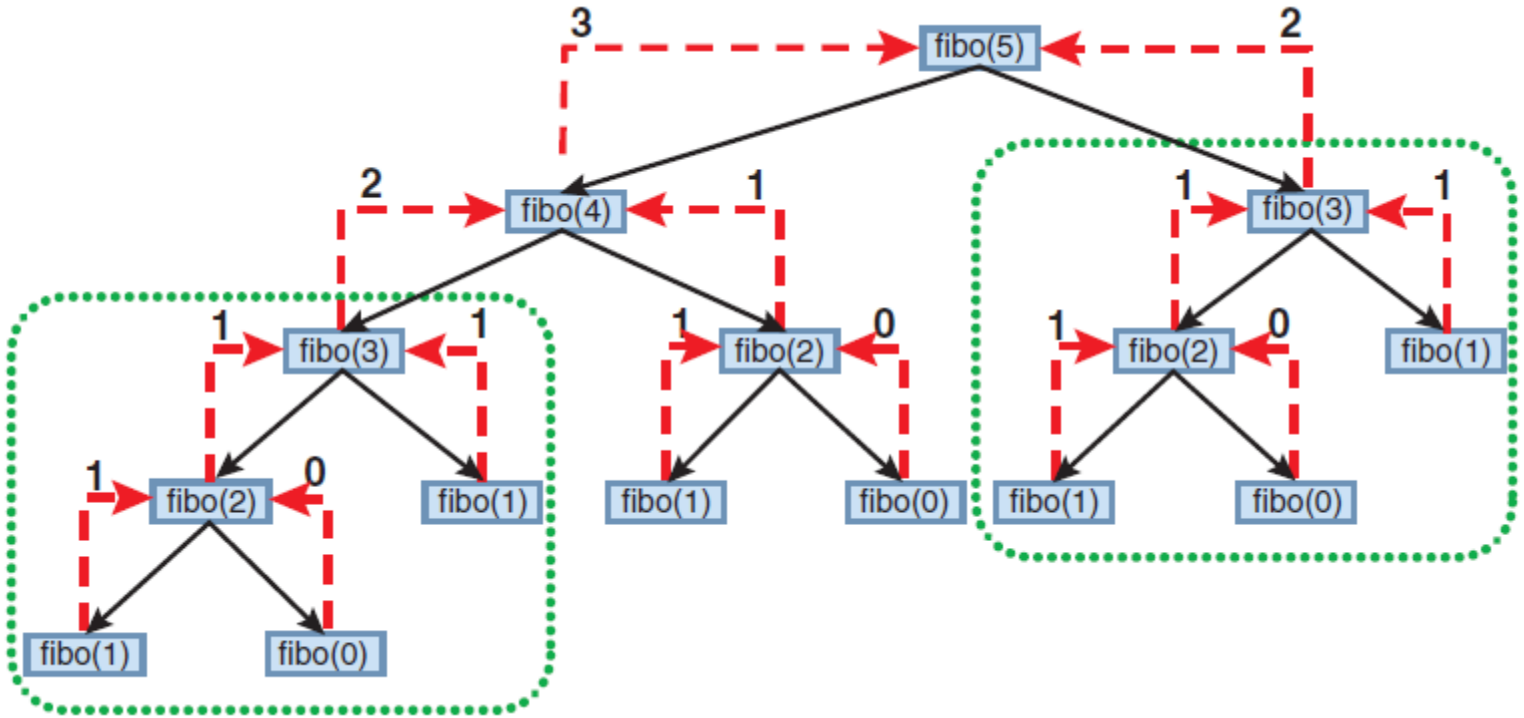
- ... mas como se verifica na imagem, elegância não significa eficiência



PUC Minas

Fibonacci

- Aumentando para **fibo(5)**



PUC Minas

Recursão (Exercício)

Identifique as chamadas recursivas e critérios de parada:

```java
public static int fiboR(int n){
  if(n == 0 || n == 1){
    return n;
  }
  else{
    return fiboR(n-1) + fiboR(n-2);
  }
}
```

```java
public static int fatorialR(int n){
  if(n == 0){
    return 1;
  }
  else{
    return n * fatorialR(n-1);
  }
}
```

Chamadas recursivas

PUC Minas

Recursão (Exercício)

Identifique as chamadas recursivas e critérios de parada:

```java
public static int fiboR(int n){
  if(n == 0 || n == 1){
    return n;
  }
  else{
    return fiboR(n-1) + fiboR(n-2);
  }
}
```

```java
public static int fatorialR(int n){
  if(n == 0){
    return 1;
  }
  else{
    return n * fatorialR(n-1);
  }
}
```

Critérios de parada

PUC Minas

Recursão

- Todo programa iterativo pode ser feito de forma recursiva e vice-versa;
    - Em determinadas situações, um problema é mais facilmente resolvido, utilizando-se de recursão;
- O conceito de recursividade é fundamental na Computação, bem como na Matemática (fatorial, números naturais, Fibonacci, entre outros);
- A recursividade pode ser direta ou indireta (A chama B que chama A);

PUC Minas

Recursão

- O SO usa uma pilha de recursão para armazenar o estado corrente do programa antes de cada chamada recursiva ainda não finalizada, e quando uma chamada é finalizada, então o SO recupera o estado armazenado na pilha;
  - As variáveis locais são recriadas para cada chamada recursiva;
- Na prática, é importante que se tenha um nível limitado de chamadas recursivas. Por quê?

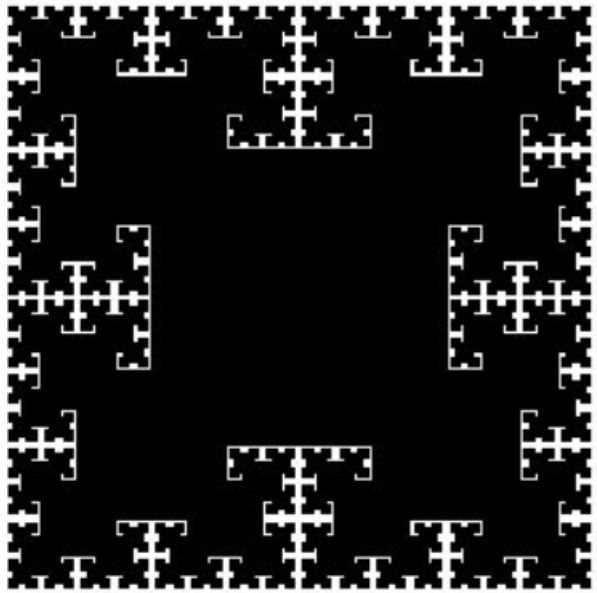
Fibonacci (Recursivo x Iterativo)

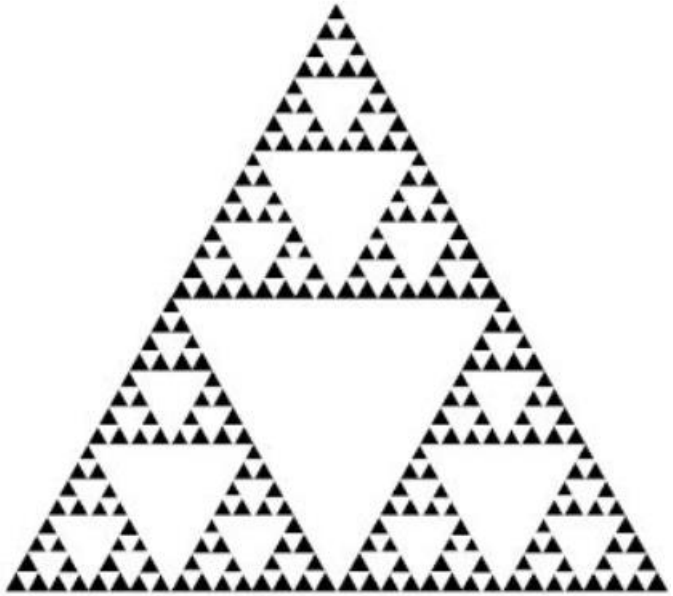
- Comparação de tempo das versões recursivas e iterativas da sequência de Fibonacci:

n	20	30	50	100
Recursiva	1 s	2 min	21 dias	10^9 anos
Iterativa	1/3 ms	1/2 ms	3/4 ms	1,5 ms

Recursão

- Um outro exemplo de recursividade são os fractais, que são padrões geométricos, que ao serem repetidos recursivamente, criam figuras interessantes;

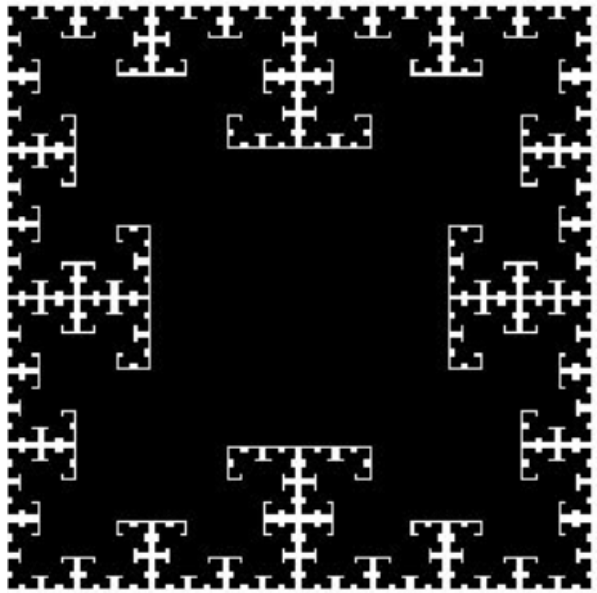


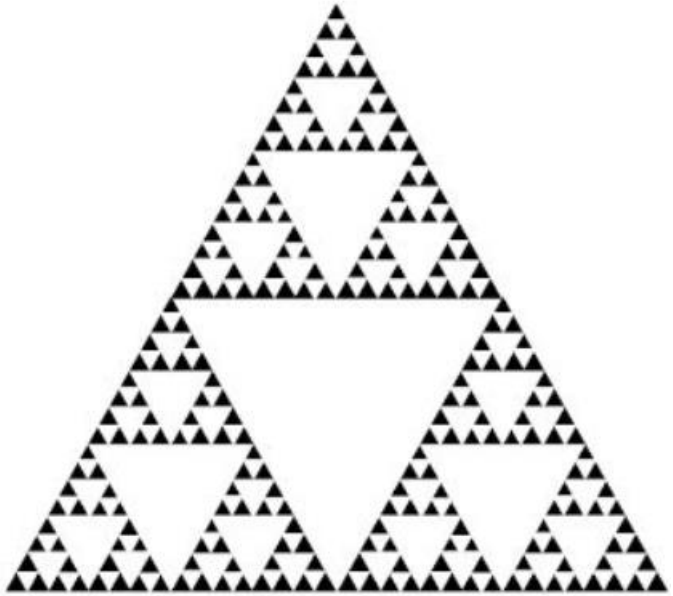


PUC Minas

Recursão

- Um outro exemplo de recursividade são os fractais, que são padrões geométricos, que ao serem repetidos recursivamente, criam figuras interessantes;





PUC Minas

Recursão (Exercício 01)

- Crie um método recursivo que receba um número inteiro positivo N e calcule o somatório dos números de 1 a N.

PUC Minas

Recursão (Exercício 01)

- Crie um método recursivo que receba um número inteiro positivo N e calcule o somatório dos números de 1 a N.

```
static int Somar(int n)
{
        if(n == 1)
            return 1;
        else
            return n + Somar(n-1);
}
```

PUC Minas

Recursão (Exercício 02)

- Faça um método recursivo que receba dois números inteiros positivos e retorne a multiplicação do primeiro pelo segundo fazendo somas.

Recursão (Exercício 02)

- Faça um método recursivo que receba dois números inteiros positivos e retorne a multiplicação do primeiro pelo segundo fazendo somas.

```
static int Multiplicar(int a, int b){
        if(b == 0)
            return 0;
        else
            return a + Multiplicar(a, b - 1);
}
```

Recursão (Exercício 03)

- Faça um método recursivo que receba um número inteiro positivo N e imprima todos os números naturais de 0 até N em ordem crescente.

Recursão (Exercício 03)

- Faça um método recursivo que receba um número inteiro positivo N e imprima todos os números naturais de 0 até N em ordem crescente.

```
static void ImprimirRecursivo(int n){
        if(n == 0)
            Console.WriteLine(n);
        else{
            ImprimirRecursivo(n-1);
            Console.WriteLine(n);
        }
}

static void Main(string[] args) {
        ImprimirRecursivo(6);
}
```

PUC Minas

Recursão (Exercício 04)

- Faça um método recursivo que receba um vetor como parâmetro, e retorne o maior elemento do vetor.

PUC Minas

Recursão (Exercício 04)

- Faça um método recursivo que receba um vetor como parâmetro, e retorne o maior elemento do vetor.

```
static int ObterMaximo(int[] v, int n){
        if (n == 1)
           return v[0];
        else {
           int x;
           x = ObterMaximo(v, n-1); //x é o máximo de v[0..n-2]
           if (x > v[n-1])
             return x;
           else
             return v[n-1];
        }
}

static void Main(string[] args) {
         int[] vet = {5,3,11,33,22,1,5};
         Console.WriteLine(ObterMaximo(vet, vet.Length));
}
```

Recursão (Exercício 04)

- Faça um método recursivo que receba um vetor como parâmetro, e retorne o maior elemento do vetor.

```
static int ObterMaximo(int[] v, int n){
        if (n == 1)
           return v[0];
        else {
           int x;
           x = ObterMaximo(v, n-1); //x é o máximo de v[0..n-2]
           if (x > v[n-1])
             return x;
           else
             return v[n-1];
        }
}

static void Main(string[] args) {
         int[] vet = {5,3,11,33,22,1,5};
         Console.WriteLine(ObterMaximo(vet, vet.Length));
}
```

Referências

- BACKES ,André. Linguagem C: completa e descomplicada. Rio de Janeiro: Elsevier, 2013.

- ZIVIANI, Nivio.PROJETO DE ALGORITMOS COM IMPLEMENTAÇÕES EM JAVA EC+. Cengage Learning Edições Ltda., 2010.

- Material da prof.ª. Ana Paula Carvalho.

- Material do prof. Daniel Capanema.

PUC Minas

PUC Minas